**LAB 20: CONSULTAS EN SQL**

**Consulta de una tabla completa**

**Álgebra relacional**

materiales

**SQL**

SELECT \*

FROM materiales

**Interfaz de usuario gráfica

Descripción generada automáticamente**

**Selección**

**Álgebra relacional**

SL{clave=1000}(materiales)

**SQL**

SELECT \*

FROM materiales

WHERE clave=1000

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**Proyección**

**Álgebra relacional**

PR{clave,rfc,fecha}(entregan)

**SQL**

SELECT clave, rfc, fecha

FROM entregan

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico, Sitio web

Descripción generada automáticamente**

**Reunión Natural**

**Álgebra relacional**

Entregan JN materiales

**SQL**

SELECT \*

FROM materiales, entregan

WHERE materiales.clave= entregan.clave

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**Si algún material no ha se ha entregado ¿Aparecería en el resultado de esta consulta?** No, porque para hacer eso, tendrías que haber hecho una diferencia entre los materiales que existen menos los materiales entregados, para ver aquellos que no se han entregado.

**Reunión con criterio específico**

**Álgebra relacional**

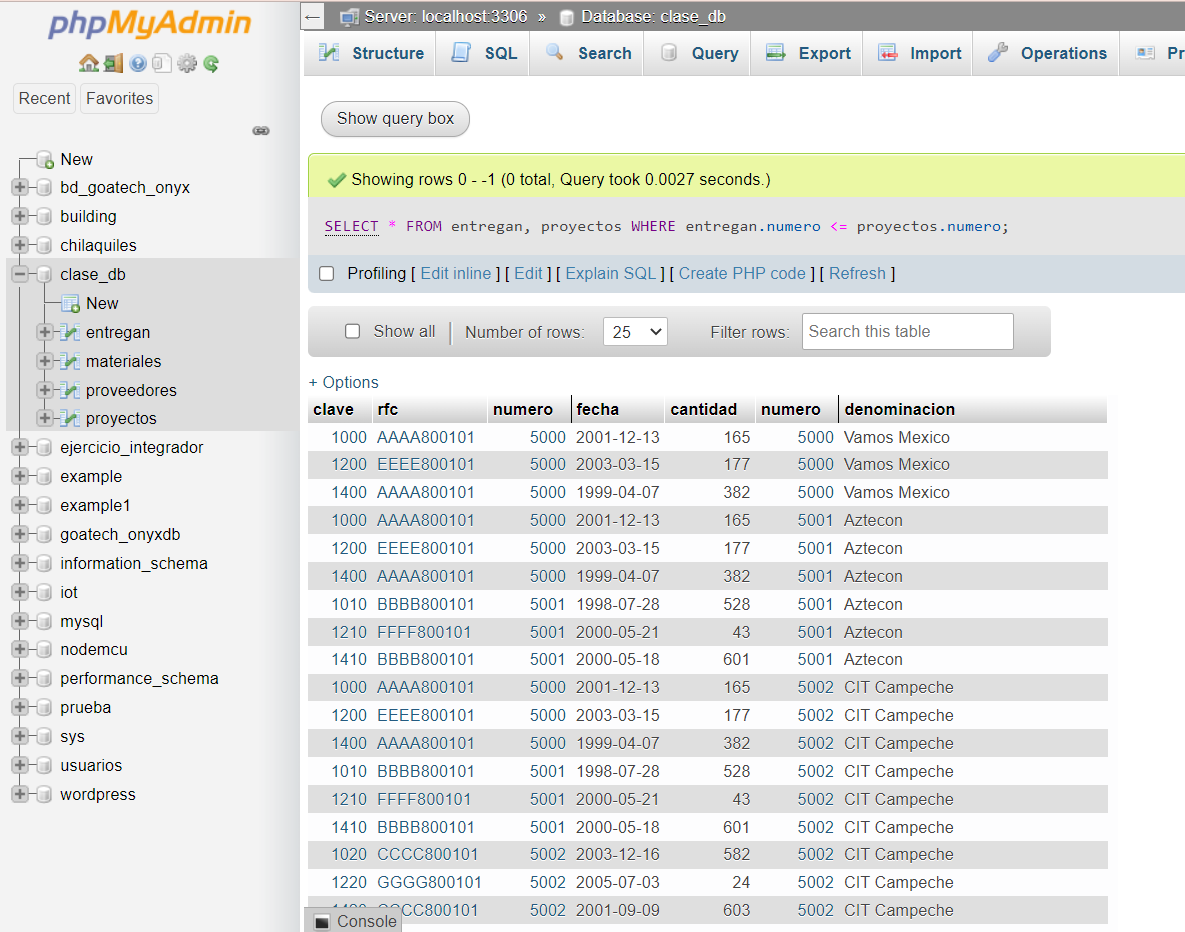
Entregan JN{entregan.numero <= proyectos.numero} proyectos

**SQL**

SELECT \*

FROM entregan, proyectos

WHERE entregan.numero <= proyectos.numero

****

**Unión (se ilustra junto con selección)**

**Álgebra relacional**

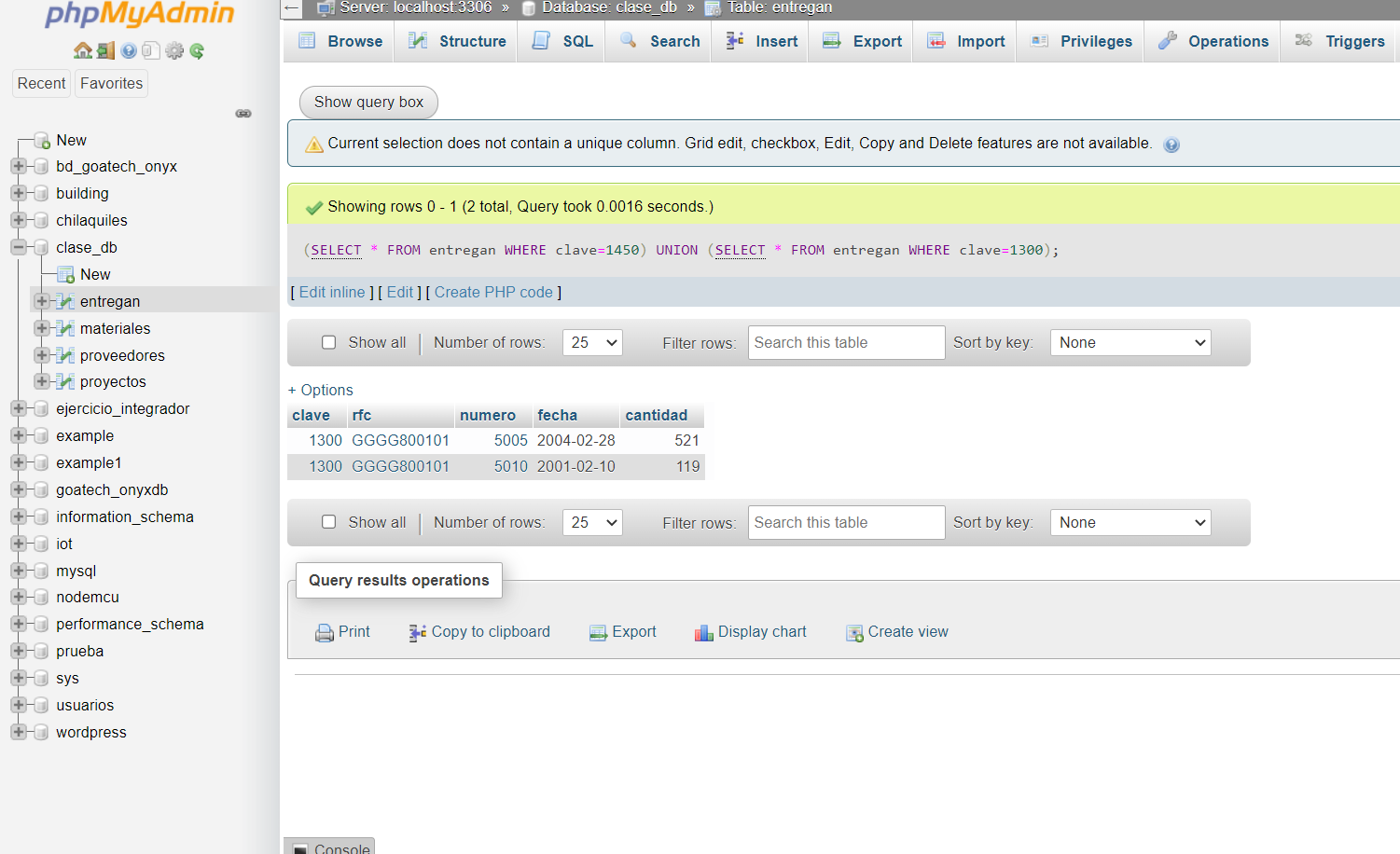
SL{clave=1450}(entregan) UN SL{clave=1300}(entregan)

**SQL**

(SELECT \* FROM entregan WHERE clave=1450)

UNION

(SELECT \* FROM entregan WHERE clave=1300)



**¿Cuál sería una consulta que obtuviera el mismo resultado sin usar el operador Unión? Compruébalo.**

SELECT \* FROM entregan WHERE clave=1450 OR clave=1300

**Intersección (se ilustra junto con selección y proyección)**

**Álgebra relacional**

PR{clave}(SL{numero=5001}(entregan)) IN PR{clave}(SL{numero=5018}(entregan))

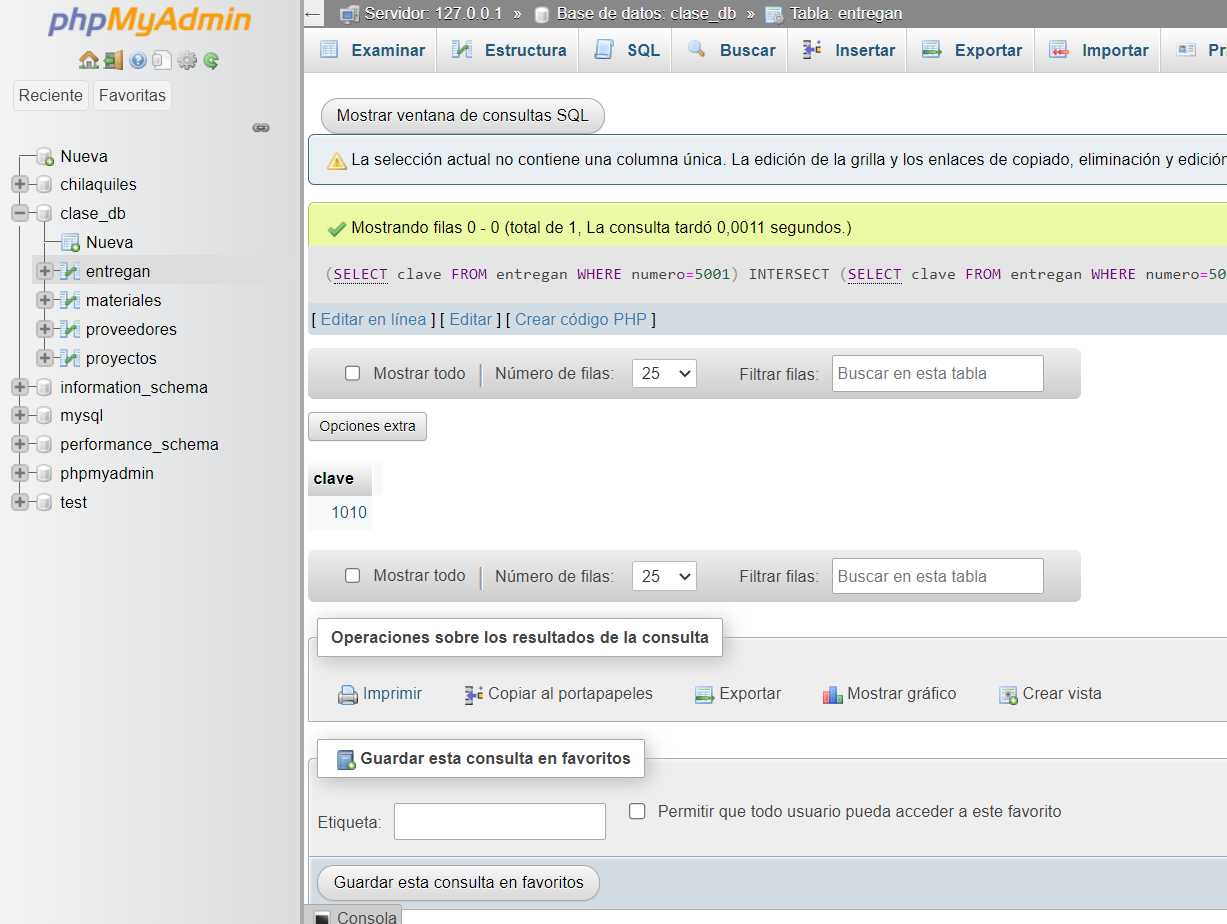
**SQL**

**Nota:** Debido a que en SQL server no tiene definida alguna palabra reservada que nos permita hacer esto de una manera entendible, veremos esta sección en el siguiente laboratorio con el uso de Subconsultas. Un ejemplo de un DBMS que si tiene la implementación de una palabra reservada para esta función es Oracle, en él si se podría generar la consulta con una sintaxis como la siguiente:

(SELECT clave FROM entregan WHERE numero=5001)

INTERSECT

(SELECT clave FROM entregan WHERE numero=5018)



**Diferencia**

**Álgebra relacional**

entregan - SL{clave=1000}(entregan)

**SQL**

(SELECT \* FROM entregan)

MINUS

(SELECT \* FROM entregan WHERE clave=1000)

Nuevamente, "minus" es una palabra reservada que no está definida en SQL Server, define una consulta que regrese el mismo resultado.

SELECT \* FROM entregan WHERE clave NOT IN (1000)

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

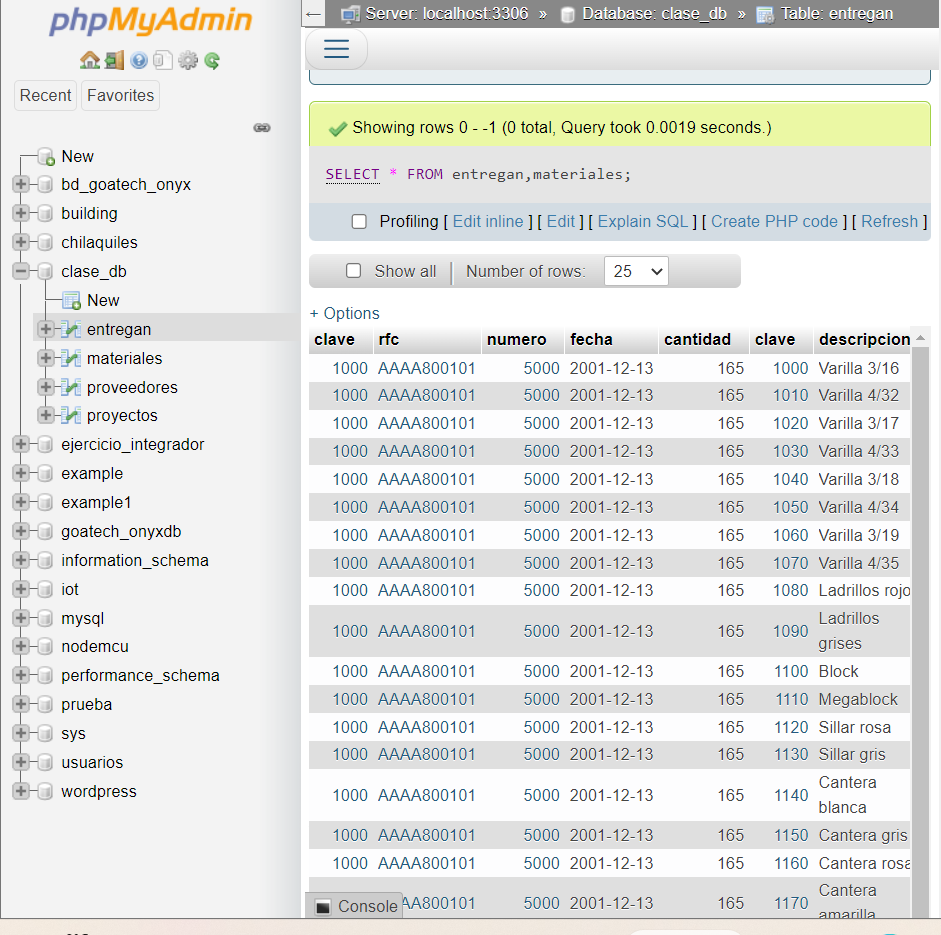
**Producto cartesiano**

**Álgebra relacional**

entregan X materiales

**SQL**

SELECT \* FROM entregan,materiales



**¿Cómo está definido el número de tuplas de este resultado en términos del número de tuplas de entregan y de materiales?** El número total de tuplas en producto cartesiano es el # de tuplas de materiales por el # de tuplas de entregan (AxB).

**Construcción de consultas a partir de una especificación**

Plantea ahora una consulta para obtener las descripciones de los materiales entregados en el año 2000.

Recuerda que la fecha puede indicarse como '01-JAN-2000' o '01/01/00'.

**Importante:** Recuerda que cuando vayas a trabajar con fechas, antes de que realices tus consultas debes ejecutar la instrucción "set dateformat dmy". Basta con que la ejecutes una sola vez para que el manejador sepa que vas a trabajar con ese formato de fechas.

SELECT descripcion

FROM materiales, entregan

WHERE materiales.clave = entregan.clave

AND entregan.fecha BETWEEN '2000-01-01' and '2001-01-01';

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

**¿Por qué aparecen varias veces algunas descripciones de material?** Porque se entregó un mismo material en varias fechas en el año 2000.

**Uso del calificador distinct**

En el resultado anterior, observamos que una misma descripción de material aparece varias veces.

Agrega la palabra distinct inmediatamente después de la palabra select a la consulta que planteaste antes.

SELECT DISTINCT descripcion

FROM materiales, entregan

WHERE materiales.clave = entregan.clave

AND entregan.fecha BETWEEN '2000-01-01' and '2001-01-01';

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**¿Qué resultado obtienes en esta ocasión?** Obtenemos los materiales cuya descripción no se repite en el año 2000.

**Ordenamientos.**

Si al final de una sentencia select se agrega la cláusula

order by campo [desc] [,campo [desc] ...]

donde las partes encerradas entre corchetes son opcionales (los corchetes no forman parte de la sintaxis), los puntos suspensivos indican que pueden incluirse varios campos y la palabra desc se refiere a descendente. Esta cláusula permite presentar los resultados en un orden específico.

Obtén los números y denominaciones de los proyectos con las fechas y cantidades de sus entregas, ordenadas por número de proyecto, presentando las fechas de la más reciente a la más antigua.

SELECT p.numero, p.denominacion, e.fecha, e.cantidad

FROM Proyectos p, entregan e

WHERE e.numero=p.numero

ORDER BY p.numero, e.fecha DESC

Interfaz de usuario gráfica

Descripción generada automáticamente

**Uso de expresiones.**

En álgebra relacional los argumentos de una proyección deben ser columnas. Sin embargo, en una sentencia SELECT es posible incluir expresiones aritméticas o funciones que usen como argumentos de las columnas de las tablas involucradas o bien constantes. Los operadores son:

+ Suma

- Resta

\* Producto

/ División

Las columnas con expresiones pueden renombrarse escribiendo después de la expresión un alias que puede ser un nombre arbitrario; si el alias contiene caracteres que no sean números o letras (espacios, puntos etc.) debe encerrarse entre comillas dobles (" nuevo nombre" ). Para SQL Server también pueden utilizarse comillas simples.

**Operadores de cadena (LIKE)**

El operador LIKE se aplica a datos de tipo cadena y se usa para buscar registros, es capaz de hallar coincidencias dentro de una cadena bajo un patrón dado.

También contamos con el operador comodín (%), que coincide con cualquier cadena que tenga cero o más caracteres. Este puede usarse tanto de prefijo como sufijo.

SELECT \* FROM materiales where Descripcion LIKE 'Si%';

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**¿Qué resultado obtienes?** Se obtiene la tabla completa de materiales donde la descripción comience con ‘Si’.

**Explica que hace el símbolo '%'.** Hace un patrón que ignora los valores de 0 o más caracteres.

**¿Qué sucede si la consulta fuera : LIKE 'Si' ?** Da todas las descripciones de los materiales que sean ‘Si’.

**¿Qué resultado obtienes?** Nada, porque no hay ni un solo material cuya descripción sea únicamente ‘Si’.

**Explica a qué se debe este comportamiento.** Esto se debe debido a que el módulo hace una búsqueda por patrones, entonces si en la consulta le pones LIKE ‘Patrón’ te va a arrojar consultas que sigan el patrón definido, pero si no especificas el patrón y únicamente poner la palabra, va a buscar entradas con esa palabra exacta, pues no es búsqueda con patrones.

**Operadores de cadena (CONCATENACIÓN)**

Otro operador de cadenas es el de concatenación, (+, +=) este operador concatena dos o más cadenas de caracteres.

Su sintaxis es : Expresión + Expresión.

Un ejemplo de su uso, puede ser: Un ejemplo de su uso, puede ser:

SELECT (Apellido + ', ' + Nombre) as Nombre FROM Personas;

DECLARE @foo varchar(40);

DECLARE @bar varchar(40);

SET @foo = '¿Que resultado';

SET @bar = ' ¿¿¿??? '

SET @foo += ' obtienes?';

PRINT @foo + @bar;

**¿Qué resultado obtienes de ejecutar el siguiente código?** No hay ningún código que se pueda correr para hacer consultas, por ende, no se da ningún resultado.

**¿Para qué sirve DECLARE?** Es para hacer una variable local.

**¿Cuál es la función de @foo?** Definir una variable varchar(40)

**¿Que realiza el operador SET?** Sirve para ponerle un valor a una variable.

**Operadores de cadena ([], [^] y \_).**

[ ] - Busca coincidencia dentro de un intervalo o conjunto dado. Estos caracteres se pueden utilizar para buscar coincidencias de patrones como sucede con LIKE.

[^] - En contra parte, este operador coincide con cualquier caracter que no se encuentre dentro del intervalo o del conjunto especificado.

\_ - El operador \_ o guion bajo, se utiliza para coincidir con un caracter de una comparación de cadenas.

Ahora explica el comportamiento, función y resultado de cada una de las siguientes consultas:

**SELECT RFC FROM Entregan WHERE RFC LIKE '[A-D]%';**

Muestra las RFC de entregan que comiencen con A,B,C o D (rango)% patrón.

**SELECT RFC FROM Entregan WHERE RFC LIKE '[^A]%';**

Muestra las RFC de entregan donde la RFC no comience con A.

**SELECT Numero FROM Entregan WHERE Numero LIKE '\_\_\_6';**

Muestra el número de entregan donde se muestren todos los número de 4 cifras que terminen con 6.

**Operadores compuestos.**

Los operadores compuestos ejecutan una operación y establecen un valor.

+ = (Suma igual)

- = (Restar igual)

\* = (Multiplicar igual)

/ = (Dividir igual)

% = (Módulo igual)

**Operadores Lógicos.**

Los operadores lógicos comprueban la verdad de una condición, al igual que los operadores de comparación, devuelven un tipo de dato booleano (True, false o unknown).

**ALL** Es un operador que compara un valor numérico con un conjunto de valores representados por un subquery. La condición es verdadera cuando todo el conjunto cumple la condición.

**ANY** o **SOME** Es un operador que compara un valor numérico con un conjunto de valores. La condición es verdadera cuando al menos un dato del conjunto cumple la condición

La sintaxis para ambos es: valor\_numerico {operador de comparación} subquery

**BETWEEN** Es un operador para especificar intervalos. Una aplicación muy común de dicho operador son intervalos de fechas.

SELECT Clave,RFC,Numero,Fecha,Cantidad

FROM Entregan

WHERE Numero Between 5000 and 5010;

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**¿Cómo filtrarías rangos de fechas?** En la parte de WHERE se pone un BETWEEN entre las fechas deseadas.

**EXISTS** Se utiliza para especificar dentro de una subconsulta la existencia de ciertas filas.

SELECT RFC,Cantidad, Fecha,Numero

FROM Entregan

WHERE Numero Between 5000 and 5010 AND

Exists (SELECT RFC

FROM Proveedores

WHERE RazonSocial LIKE 'La%' and Entregan.RFC = Proveedores.RFC);

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

**¿Qué hace la consulta?** Selecciona rfc, cantidad, fecha y número de entregan donde el número está entre 500 y 5010 y que además su razón social empiece con ‘La’.

**¿Qué función tiene el paréntesis ( ) después de EXISTS?** Es para indicar que existe una subconsulta la cual se debe de cumplir también como parte de la solución.

**IN** Especifica si un valor dado tiene coincidencias con algún valor de una subconsulta. **NOTA:** Se utiliza dentro del WHERE pero debe contener un parametro. Ejemplo: Where proyecto.id IN Lista\_de\_Proyectos\_Subquery

Tomando de base la consulta anterior del EXISTS, realiza el query que devuelva el mismo resultado, pero usando el operador IN

SELECT RFC,Cantidad, Fecha,Numero

FROM Entregan

WHERE Numero Between 5000 and 5010 AND

RFC IN (SELECT RFC

FROM Proveedores

WHERE RazonSocial LIKE 'La%');

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**NOT** Simplemente niega la entrada de un valor booleano.

Tomando de base la consulta anterior del EXISTS, realiza el query que devuelva el mismo resultado, pero usando el operador NOT IN Realiza un ejemplo donde apliques algún operador : ALL, SOME o ANY.

SELECT RFC,Cantidad, Fecha,Numero

FROM Entregan

WHERE Numero Between 5000 and 5010 AND

RFC NOT IN (SELECT RFC

FROM Proveedores

WHERE RazonSocial NOT LIKE 'La%' and Entregan.RFC = Proveedores.RFC); Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

El Operador **TOP**, es un operador que recorre la entrada, un query, y sólo devuelve el primer número o porcentaje especifico de filas basado en un criterio de ordenación si es posible.

**¿Qué hace la siguiente sentencia? Explica por qué.**

SELECT TOP 2 \* FROM Proyectos

Regresaría los primeros dos renglones de la tabla proyectos, pues TOP indica primer, y el dos quiere indicar que los primeros dos.

**¿Qué sucede con la siguiente consulta? Explica por qué.**

SELECT TOP Numero FROM Proyectos

Número es una columna de proyectos, por lo que no se puede poner esta para indicar el número de renglones ya que no es un número. Regresa error.

**Modificando la estructura de un tabla existente.**

Agrega a la tabla materiales la columna PorcentajeImpuesto con la instrucción:

ALTER TABLE materiales ADD PorcentajeImpuesto NUMERIC(6,2);

A fin de que los materiales tengan un impuesto, les asignaremos impuestos ficticios basados en sus claves con la instrucción:

UPDATE materiales SET PorcentajeImpuesto = 2\*clave/1000;

esto es, a cada material se le asignará un impuesto igual al doble de su clave dividida entre diez.

Revisa la tabla de materiales para que compruebes lo que hicimos anteriormente.

**¿Qué consulta usarías para obtener el importe de las entregas es decir, el total en dinero de lo entregado, basado en la cantidad de la entrega y el precio del material y el impuesto asignado?**

SELECT e.Cantidad, m.Precio, e.Numero, m.Descripcion, m.PorcentajeImpuesto,

e.Cantidad \* m.Precio \* (1 + m.PorcentajeImpuesto / 100) AS Importe

FROM materiales m, entregan e

WHERE e.clave = m.clave;

Interfaz de usuario gráfica

Descripción generada automáticamente

**Creación de vistas**

La sentencia:

Create view nombrevista (nombrecolumna1 , nombrecolumna2 ,..., nombrecolumna3 )

as select...

Permite definir una vista. Una vista puede pensarse como una consulta etiquetada con un nombre, ya que en realidad al referirnos a una vista el DBMS realmente ejecuta la consulta asociada a ella, pero por la cerradura del álgebra relacional, una consulta puede ser vista como una nueva relación o tabla, por lo que es perfectamente válido emitir la sentencia

select \* from nombrevista

¡Como si nombrevista fuera una tabla!

Comprueba lo anterior, creando vistas para cinco de las consultas que planteaste anteriormente en la práctica . Posteriormente revisa cada vista creada para comprobar que devuelve el mismo resultado.

CREATE VIEW Entregados2000

AS SELECT descripcion

FROM materiales, entregan

WHERE materiales.clave = entregan.clave

AND entregan.fecha BETWEEN '2000-01-01' and '2001-01-01';

(SELECT \* FROM Entregados2000)

CREATE VIEW ProyectosRZ

AS SELECT RFC,Cantidad, Fecha,Numero

FROM Entregan

WHERE Numero Between 5000 and 5010 AND

RFC IN (SELECT RFC

FROM Proveedores

WHERE RazonSocial LIKE 'La%');

(SELECT \* FROM ProyectosRZ)

CREATE VIEW MatPrecImp

AS SELECT e.Cantidad, m.Precio, e.Numero, m.Descripcion, m.PorcentajeImpuesto,

e.Cantidad \* m.Precio \* (1 + m.PorcentajeImpuesto / 100) AS Importe

FROM materiales m, entregan e

WHERE e.clave = m.clave;

(SELECT \* FROM MatPrecImp)

CREATE VIEW DescrSi

AS SELECT \* FROM materiales where Descripcion LIKE 'Si%';

(SELECT \* FROM DescrSi)

CREATE VIEW EntregasNot1000

AS SELECT \* FROM entregan WHERE clave NOT IN (1000);

(SELECT \* FROM EntregasNot1000)

**Reporte**

A continuación, se te dan muchos enunciados de los cuales deberás generar su correspondiente consulta.

En el reporte incluye la sentencia, una muestra de la salida (dos o tres renglones) y el número de renglones que SQL Server reporta al final de la consulta.

1. **Los materiales (clave y descripción) entregados al proyecto "México sin ti no estamos completos".**

SELECT m.clave, m.descripcion

FROM materiales m, entregan e, proyectos pr

WHERE m.clave=e.clave

AND pr.numero=e.numero

AND pr.denominacion= 'Mexico sin ti no estamos completos';

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Renglones: 3

1. **Los materiales (clave y descripción) que han sido proporcionados por el proveedor "Acme tools".**

SELECT m.clave, m.descripcion

FROM materiales m, entregan e, proveedores p

WHERE p.rfc=e.rfc

AND m.clave=e.clave

AND p.razonsocial='Acme tools';



Renglones: 0

1. **El RFC de los proveedores que durante el 2000 entregaron en promedio cuando menos 300 materiales.**

SELECT p.rfc

FROM proveedores p, entregan e

WHERE p.rfc = e.rfc

AND e.fecha BETWEEN '2000-01-01' AND '2001-01-01'

GROUP BY p.rfc

HAVING SUM(e.cantidad) >= 300;

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Renglones: 4

1. **El Total entregado por cada material en el año 2000.**

SELECT m.descripcion, SUM(e.cantidad)

FROM entregan e, materiales m

WHERE e.clave=m.clave

AND e.fecha BETWEEN '2000-01-01' AND '2001-01-01'

GROUP BY m.clave, m.descripcion

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente

Renglones: 11

1. **La Clave del material más vendido durante el 2001. (se recomienda usar una vista intermedia para su solución)**

CREATE VIEW matMasVen

AS SELECT m.clave, m.descripcion, SUM(e.cantidad) AS TotalEnt

FROM materiales m, entregan e

WHERE m.clave=e.clave

AND e.fecha BETWEEN '2001-01-01' AND '2002-01-01'

GROUP BY m.clave, m.descripcion;

SELECT clave FROM matMasVen LIMIT 1;



Renglones: 1

1. **Productos que contienen el patrón 'ub' en su nombre.**

SELECT \*

FROM materiales

WHERE descripcion LIKE ('%ub%');

1. **Denominación y suma del total a pagar para todos los proyectos.**

SELECT pr.denominacion, SUM(e.cantidad\*m.precio\*(1 + m.porcentajeimpuesto / 100)) AS TotalPagar

FROM materiales m, proyectos pr, entregan e

WHERE m.clave=e.clave AND pr.numero=e.numero

GROUP BY denominacion

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Renglones: 20

1. **Denominación, RFC y RazonSocial de los proveedores que se suministran materiales al proyecto Televisa en acción que no se encuentran apoyando al proyecto Educando en Coahuila (Solo usando vistas).**

CREATE VIEW Televisa

AS SELECT p.rfc, p.razonsocial

FROM proveedores p, proyectos pr, entregan e

WHERE p.rfc=e.rfc AND e.numero=pr.numero

AND pr.denominacion='Televisa en accion';

CREATE VIEW ProvEd

AS SELECT p.rfc, p.razonsocial

FROM proveedores p, proyectos pr, entregan e

WHERE p.rfc=e.rfc AND e.numero=pr.numero

AND pr.denominacion='Educando en Coahuila';

SELECT \* FROM Televisa

EXCEPT

SELECT \* FROM ProvEd;

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Renglones: 2

1. **Denominación, RFC y RazonSocial de los proveedores que se suministran materiales al proyecto Televisa en acción que no se encuentran apoyando al proyecto Educando en Coahuila (Sin usar vistas, utiliza not in, in o exists).**

SELECT DISTINCT p.razonsocial, p.rfc

FROM proveedores p, entregan e, proyectos pr

WHERE p.rfc = e.rfc AND e.numero = pr.numero

AND pr.denominacion = 'Televisa en accion'

AND p.rfc NOT IN (

SELECT p.rfc

FROM proveedores p, entregan e, proyectos pr

WHERE p.rfc = e.rfc

AND e.numero = pr.numero

AND pr.denominacion = 'Educando en Coahuila');

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Renglones: 2

1. **Costo de los materiales y los Materiales que son entregados al proyecto Televisa en acción cuyos proveedores también suministran materiales al proyecto Educando en Coahuila.**

SELECT m.descripcion, SUM(e.cantidad \* m.precio \* (1 + m.porcentajeimpuesto / 100)) AS CostoMat

FROM materiales m, proyectos pr, entregan e

WHERE m.clave = e.clave AND pr.numero= e.numero

AND pr.denominacion = 'Televisa en accion'

AND e.rfc IN (

SELECT e.rfc

FROM entregan e, proyectos pr

WHERE e.numero = pr.numero

AND pr.denominacion = 'Educando en Coahuila'

)

GROUP BY m.descripcion;

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Renglones: 2

1. **Nombre del material, cantidad de veces entregados y total del costo de dichas entregas por material de todos los proyectos.**

SELECT m.descripcion, COUNT(e.cantidad) AS CantEnt,

SUM(e.cantidad \* m.precio \* (1 + m.porcentajeimpuesto / 100)) AS Costo

FROM materiales m, entregan e

WHERE m.clave = e.clave

GROUP BY m.descripcion;

Tabla

Descripción generada automáticamente

Renglones: 42